Sistemas distribuidos
Laboratorio #2

Ayudante: Maximiliano Pérez Rodríguez
Profesor: Daniel Wladdimiro Cottet

Primer semestre 2017

1 Objetivo

Aplicar los conceptos de arquitectura distribuida y caché, de tal manera de disminuir la latencia en las consultas de una red P2P.

2 Enunciado

Dentro de las redes P2P se encuentran las de tipo estructuradas, las cuales están dadas por un *overlay* determinístico, y en general, basado en figuras geométricas. De esta manera, las búsquedas son completa, eficientes y escalables. Por lo tanto, se propone construir un red P2P estructurada con forma de circunferencia.

Una de las optimizaciones que se realiza a las arquitecturas distribuidas es añadir el componente de caché a ésta. En el caso de la red P2P sirve para disminuir la cantidad de saltos para realizar la búsqueda, disminuyendo los costos y latencia de la consulta. Dado lo anterior, se propone construir una red P2P estructura donde cada *peer* esté compuesto por dos componentes: una caché y una base de datos. En la Figura 1 se ejemplifica la red descrita anteriormente, donde el *peer* morado está destacado para mostrar cuales son los componentes que tendría cada uno de los *peers* en la red. Para efectos de este laboratorio, se debe definir una base de datos de tamaño b y una caché de tamaño c, cuyos valores deben ser seteados por el archivo de configuración del programa. En el caso de la inicialización, sólo la base de datos debe realizarse, y no así el caché, cuyos valores se deben ir agregando a medida que se vayan realizando consultas en la red.

Pues bien, para realizar una consulta por parte de un *peer*, se debe enviar un mensaje donde éste contendrá el *peer* de destino y la consulta realizada, la cual queda a criterio personal[1]. Dado lo anterior, la búsqueda se realizará de la siguiente manera:

[1]Recomendación: Se podría utilizar un número natural para realizar la consulta y poblar la base de datos. Cada número corresponde a una consulta única, y ésta no se puede repetir en algún otro *peer* que no sea el dónde esté alojado.

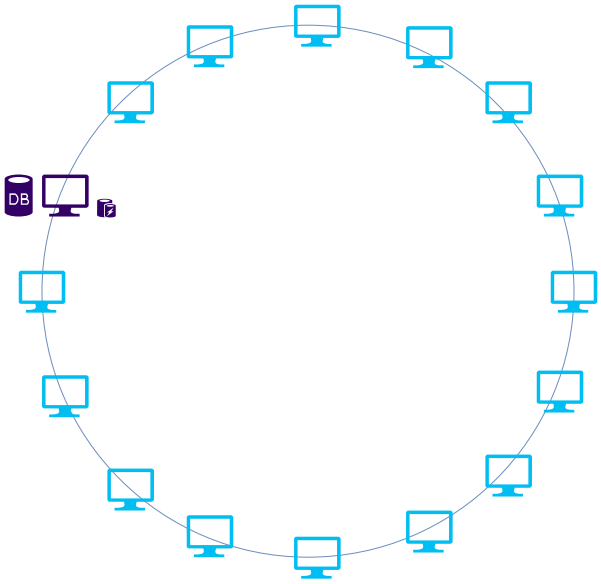

Figure 1: Ejemplo de red P2P estructurada.

```
Busqueda:
    Si el Peer_Actual posee consulta en caché
        Enviar consulta del caché
    Sino consultar por el Peer más cercano al Peer_Destino
        Si Peer más cercano es el Peer_Actual
            Consultar al próximo vecino según el sentido del reloj
         Si Peer más cercano es uno de la DHT
            Consultar al Peer definido en la DHT
```

Para efectos de este tipo de búsqueda, se va a considerar el *peer* más cercano; aquel *peer* que deba realizar menor cantidad de saltos en sentido horario a la estructura de la red.

Pues bien, en el caso que se deba consultar al *peer* más cercano, se debe analizar si éste corresponde al *peer* actual o un *peer* de la `DHT`. Para esto, se debe definir una `DHT` con tamaño $1 + 2d$, donde $d \in N$. La `DHT` debe ser creada al inicializar la red, y contendrá el ID de un *peer* y $Objeto$ de éste, de esta manera, en caso de consultar por el ID, éste retornará el *peer* para posteriormente enviar el mensaje de búsqueda a éste. Para definir la estructura de la `DHT`, se deben almacenar los *peer* que están a una distancia igual a la de $(n/2^x)$, donde n es la cantidad de *peer* en la red, con $x \in N$ y $2^x \leq n$. Los valores de d y n se deben definir por el archivo de configuración del programa.

En la Figura 2 se observa un ejemplo de la red, donde se va a analizar la `DHT` del *peer* con $ID = 13$. La configuración de esta red corresponde a $d = 1$ y $n = 16$. Al inicializar la DHT, se considero los *peers* que están a una distancia de $(n/2^1) = 16/2 = 8$ y $(n/2^2) = 16/4 = 4$ del *peer* analizado. Como se observa, los *peers* que cumplen estas condiciones son los de color verde, mientras que el morado es el *peer* analizado.

Supongamos que deseamos realizar una búsqueda desde el *peer* morado, consulta por el dato

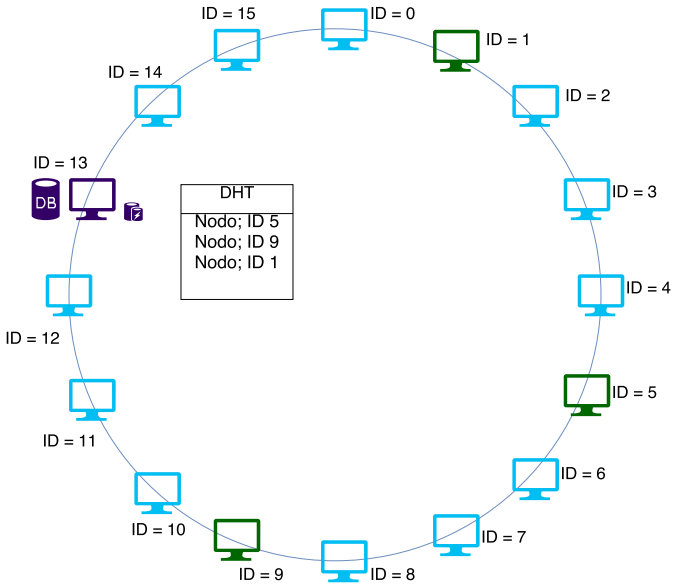

Figure 2: Ejemplo de DHT en la red P2P.

X que está en el *peer* con $ID = 11$. Dado esto, como ya se había explicado anteriormente, se debe consultar primero si se encuentra en la caché. En caso de no estarlo, se debe consultar al *peer* más cercano, considerando la `DHT` y el *peer* actual. Desde le *peer* actual la cantidad de saltos que debemos realizar para llegar al *peer* con $ID = 11$ es 14. Según desde la DHT para *peer* $ID = 1$ sería 10, *peer* $ID = 5$ sería 6 y *peer* $ID = 9$ sería 2. Por lo tanto, el *peer* que va a continuar la búsqueda será el *peer* con $ID = 9$. En la Figura 3 se observa como se realiza la búsqueda, analizando la posición del *peer* actual y los de la `DHT`.

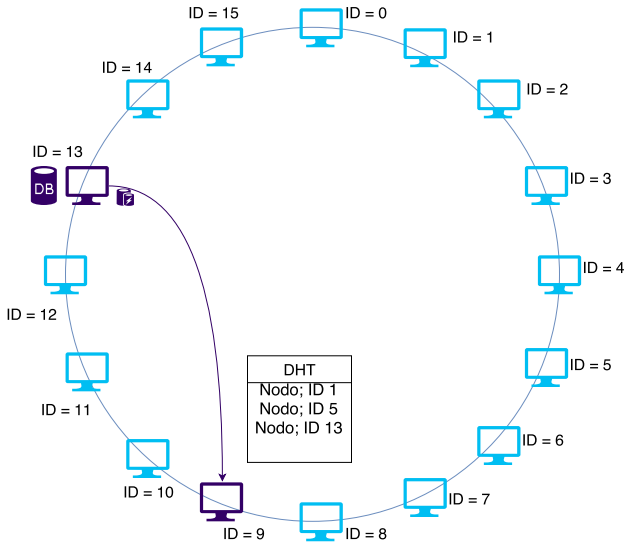

Figure 3: Ejemplo de búsqueda en la red P2P estructurada.

Ya en el *peer* con $ID = 9$ se consulta en el caché, en el caso de no existir, se analiza cuál es el *peer* más cercano. En este caso, el *peer* más cercano es el *peer* actual, por lo tanto, se consulta al próximo vecino según el sentido horario, en este caso, el *peer* con $ID = 10$ como se observa en la Figura 4.

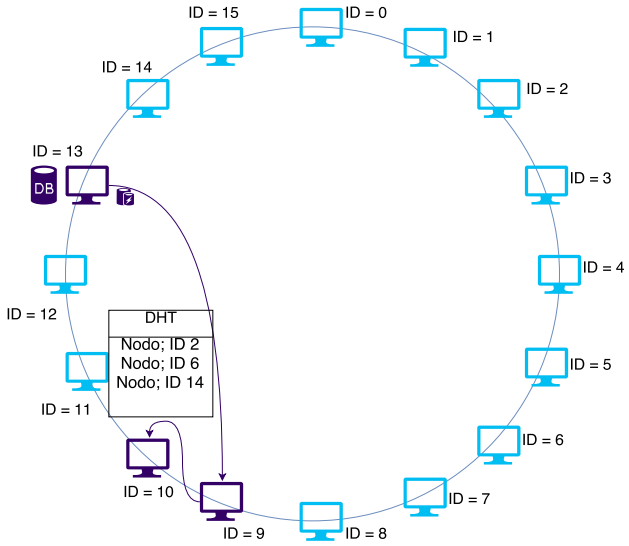

Figure 4: Ejemplo de búsqueda en la red P2P estructurada.

Dado que el *peer* más cercano al *peer* de búsqueda, es a él mismo, entonces se debe consultar por el próximo vecino en sentido horario, y en este caso, es el *peer* de la búsqueda como se observa en la Figura 5.

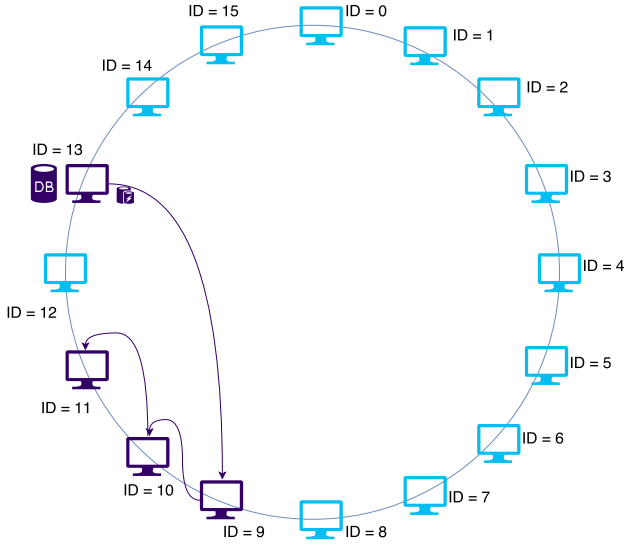

Figure 5: Ejemplo de búsqueda en la red P2P estructurada.

Como se ha podido localizar la búsqueda, se debe devolver la respuesta, de ser así, se debe insertar la consulta en el caché de los *peers* consultados, es decir, en los *peers* con $ID = \{10, 9, 13\}$. En la Figura 6 están las líneas de retorno de cada uno de los *peers* que se consultó por la pregunta, y se insertará la búsqueda en el caché.

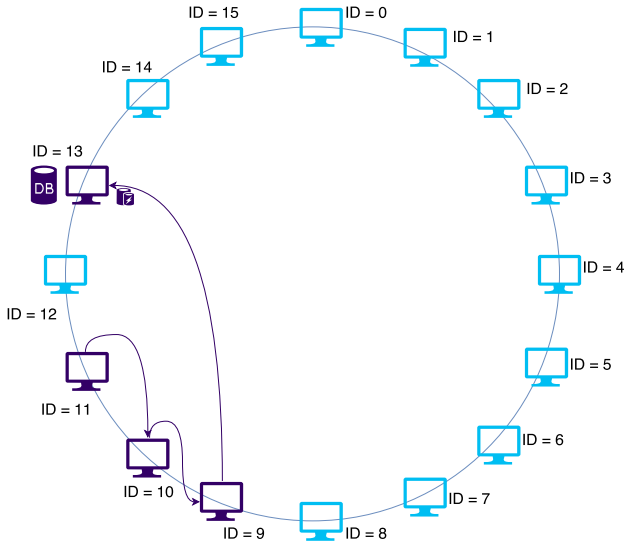

Figure 6: Ejemplo de retorno de la búsqueda en la red P2P estructurada.

3 Evaluación

La entrega del laboratorio deberá ser subida al Moodle del curso en Usachvirtual. Este debe ser subido en un archivo comprimido `tar.gz`, con su respectivo archivo de configuración y código fuente para la ejecución del simulador.

En el caso que no se cumpla con ninguna de las solicitudes descritas anteriormente, se evaluará con la nota mínima.

La *fecha de término* del laboratorio es el **1 de Junio**.

4 Importante

- Copy/Paste tendrá un descuento de 1 punto si no cuenta con el debido reconocimiento de la fuente desde donde se obtuvieron los extractos del código.
- El trabajo debe ser coherente con lo especificado en el enunciado.
- Por cada día de atraso desde la fecha de entrega del código tendrá un descuento de 1 punto.

5 Referencias

- PeerSim. `http://peersim.sourceforge.net/`
- Tanenbaum, A. & Van Steen, M. - Distributed Systems: Principles and Paradigms.